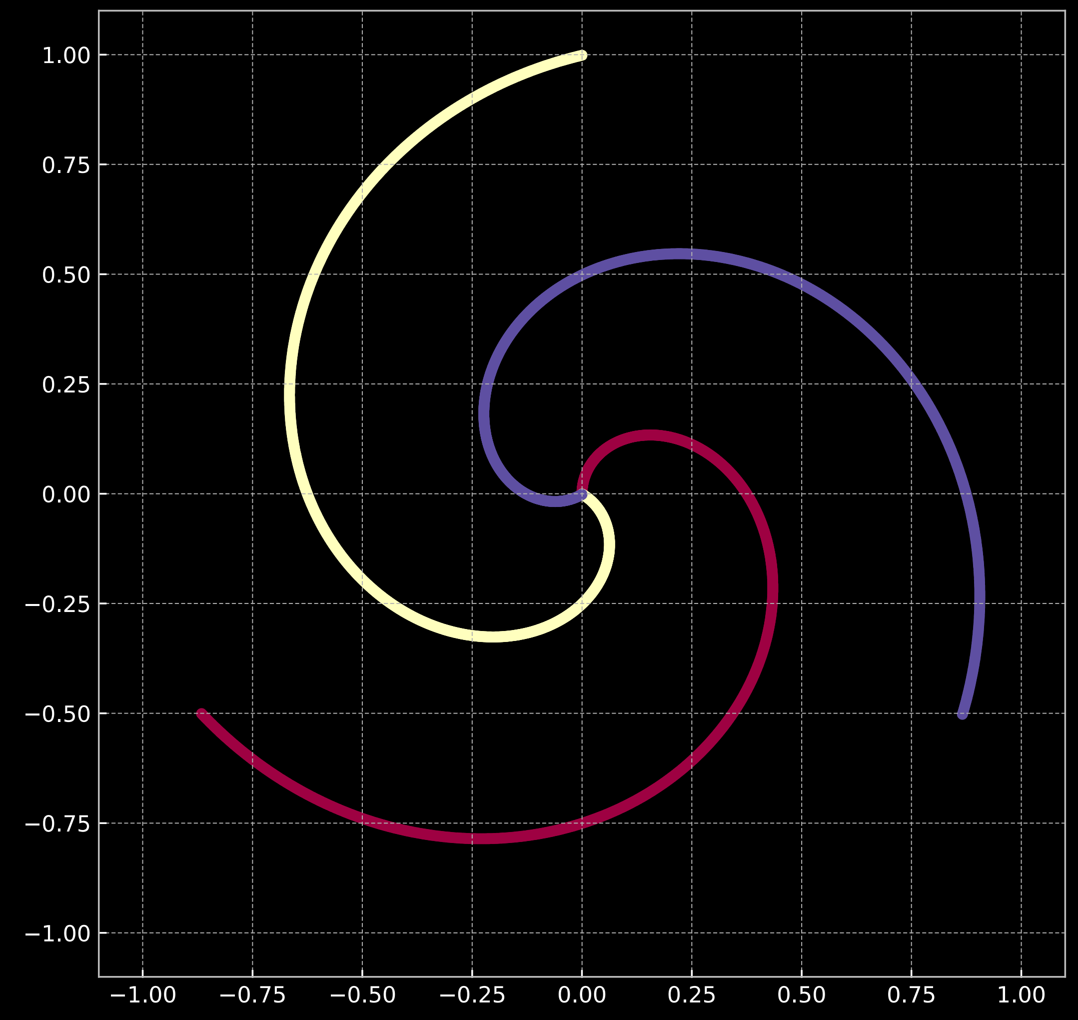
Foundations of Deep Learning

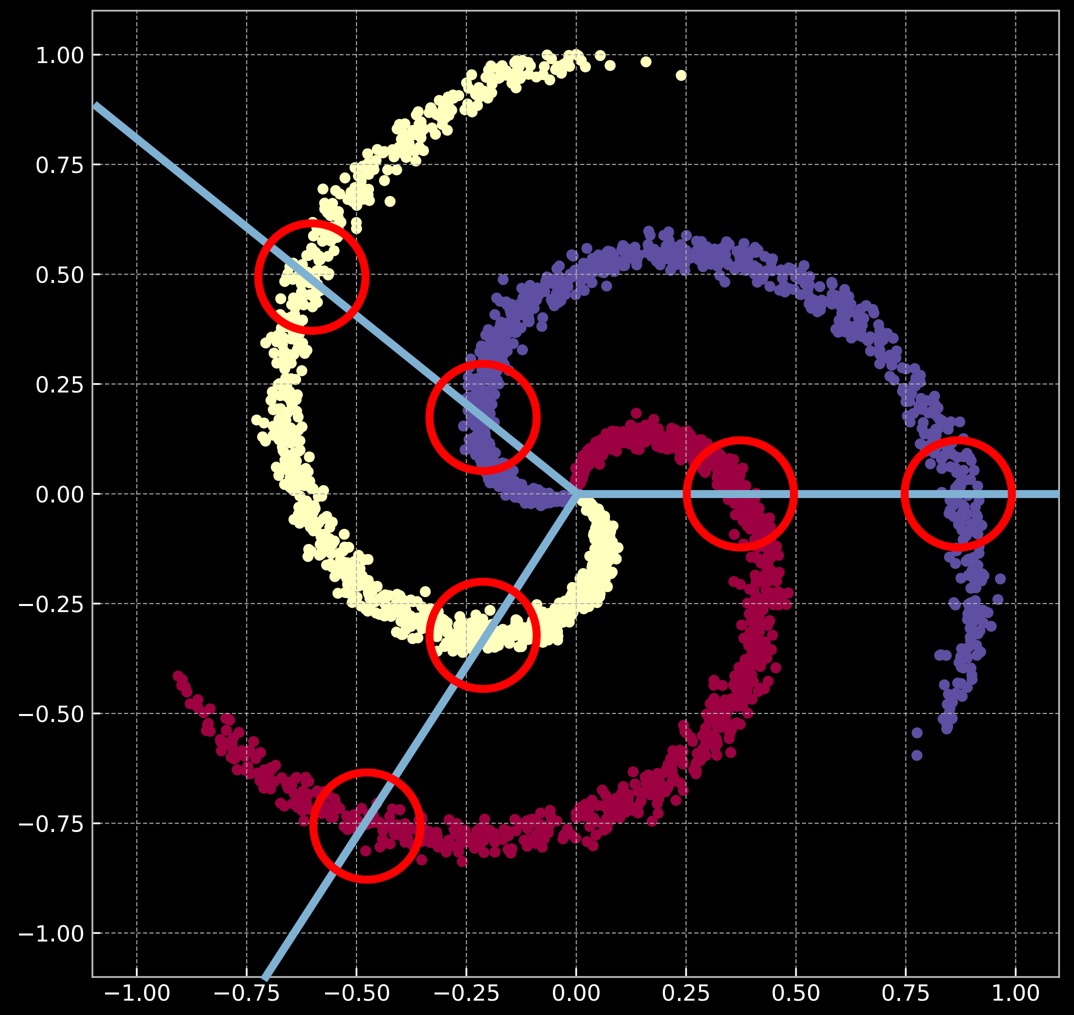
ALF

Alfredo Canziani
@alfcnz

ANN – supervised learning

Classification



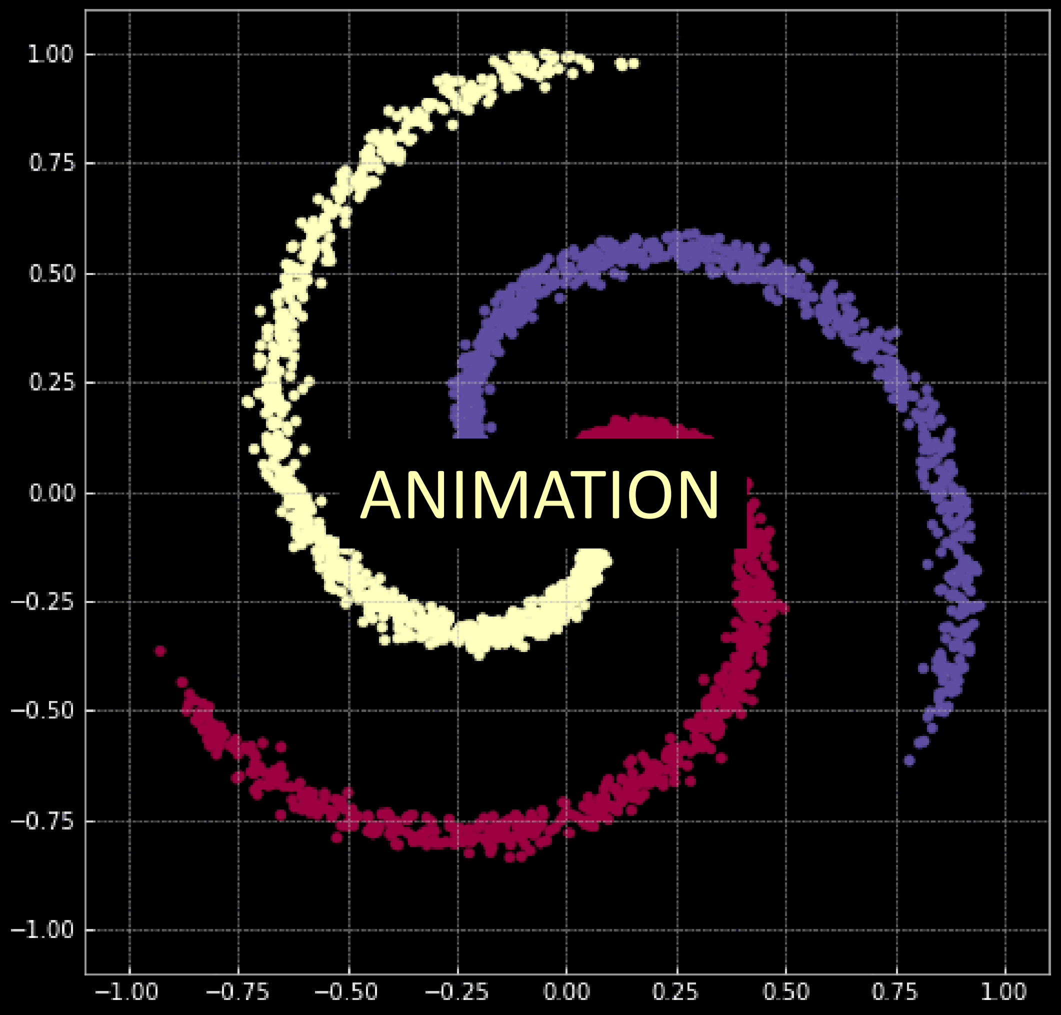


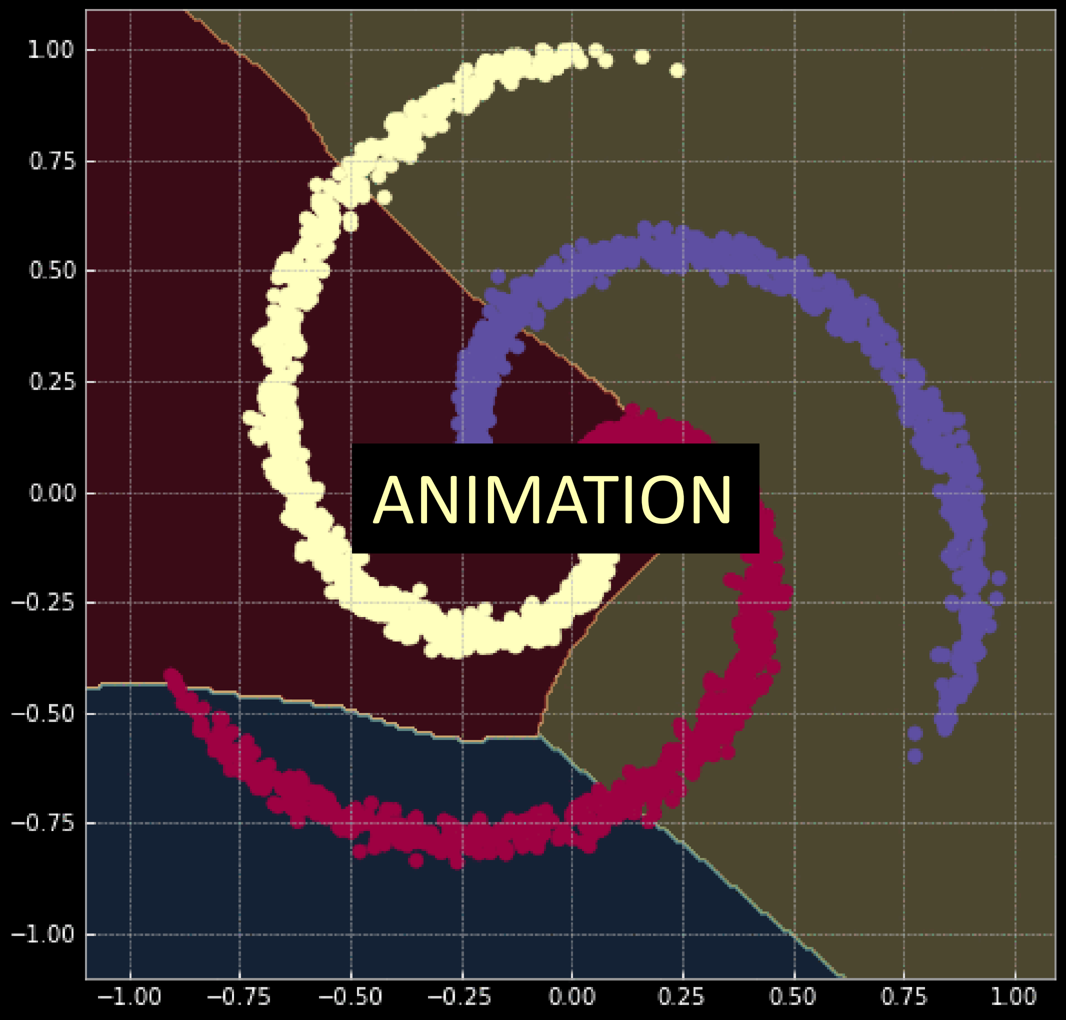
$$X_k(t) = t \begin{pmatrix} \sin\left[\frac{2\pi}{K}(2t+k-1)\right] \\ \cos\left[\frac{2\pi}{K}(2t+k-1)\right] \end{pmatrix}$$

$$0 \le t \le 1, \quad k = 1, \cdots, K$$

$$X_k(t) = t \begin{pmatrix} \sin\left[\frac{2\pi}{K}(2t+k-1)\right] \\ \cos\left[\frac{2\pi}{K}(2t+k-1)\right] \end{pmatrix}$$

$$\boxed{+\mathcal{N}(0, \sigma^2)}$$


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
ANIMATION


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
ANIMATION

Train data

$$\begin{pmatrix}1 & 0 & 0\end{pmatrix}$$
$$\begin{pmatrix}0 & 1 & 0\end{pmatrix}$$
$$\begin{pmatrix}0 & 0 & 1\end{pmatrix}$$

1-hot encoding

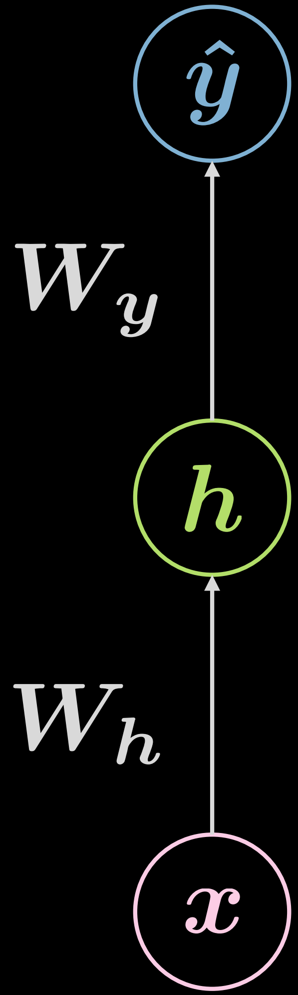
$$\boldsymbol{X} = \overbrace{\begin{bmatrix} — \boldsymbol{x}^{(1)} — \\ — \boldsymbol{x}^{(2)} — \\ \vdots \\ — \boldsymbol{x}^{(m)} — \end{bmatrix}}^{n} m \qquad \boldsymbol{Y} = \overbrace{\begin{bmatrix} — \boldsymbol{y}^{(1)} — \\ — \boldsymbol{y}^{(2)} — \\ \vdots \\ — \boldsymbol{y}^{(m)} — \end{bmatrix}}^{K} m \qquad \boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} m$$

$$\boldsymbol{x}^{(i)} \in \mathbb{R}^n \qquad \boldsymbol{y}^{(i)} \in \{0,1\}^K \qquad c_i \in \{1, 2, \cdots, K\}$$

Fully connected (FC) layer

j-th row of $W^{(1)}$

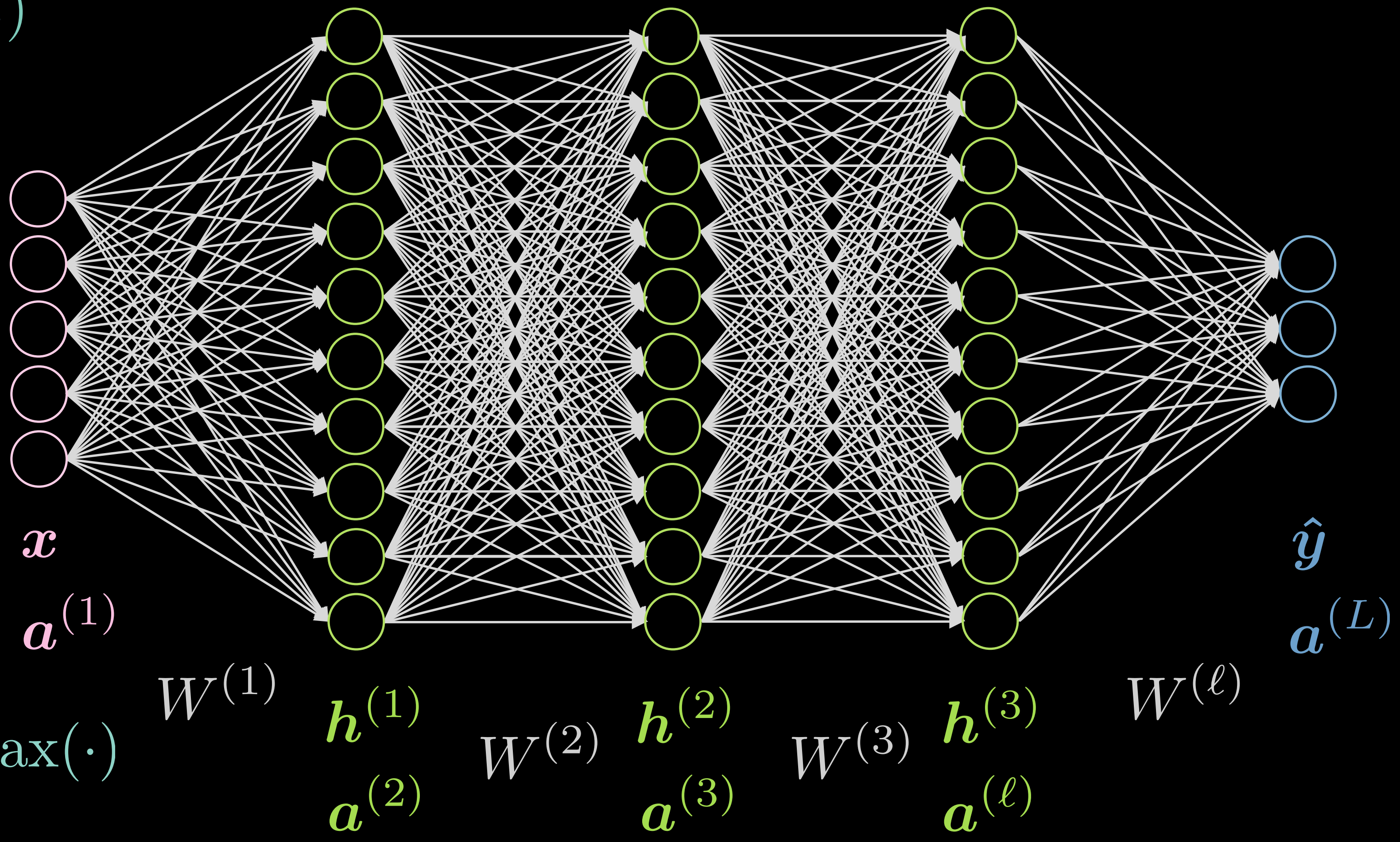
$$a_j^{(2)} = f(\boxed{\boldsymbol{w}^{(j)}}\boldsymbol{x} + b_j) = f\left(\left(\sum_{i=1}^{n} w_i^{(j)} x_i\right) + b_j\right)$$



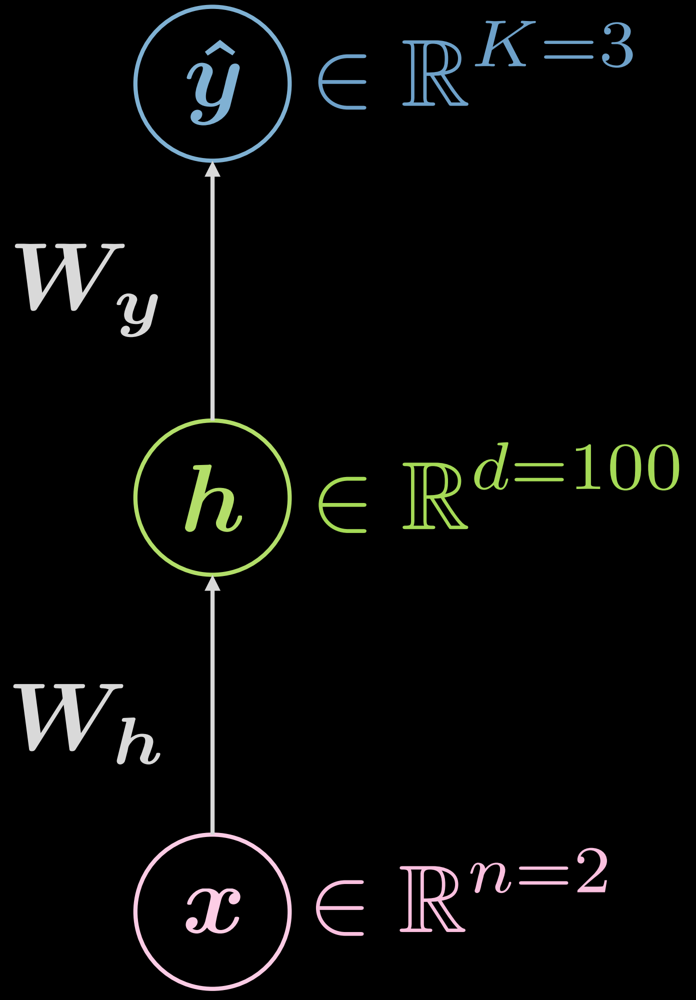
$$\boldsymbol{h} = f(\boldsymbol{W_h x} + \boldsymbol{b_h})$$

$$\hat{\boldsymbol{y}} = g(\boldsymbol{W_y h} + \boldsymbol{b_y})$$

$f, g = (\cdot)^+, \sigma(\cdot),$
$\tanh(\cdot), \mathrm{soft}(\mathit{arg})\mathrm{max}(\cdot)$



Neural network (inference)



$$\boldsymbol{h} = f(\boldsymbol{W_h}\boldsymbol{x} + \boldsymbol{b_h})$$
$$\hat{\boldsymbol{y}} = g(\boldsymbol{W_y}\boldsymbol{h} + \boldsymbol{b_y})$$

$$\boldsymbol{W_h} \in \mathbb{R}^{d \times n}$$
$$\boldsymbol{b_h} \in \mathbb{R}^{d}$$
$$\boldsymbol{W_y} \in \mathbb{R}^{K \times d}$$
$$\boldsymbol{b_y} \in \mathbb{R}^{K}$$

$$f, g = (\cdot)^+, \sigma(\cdot),$$
$$\tanh(\cdot), \mathrm{soft}(arg)\mathrm{max}(\cdot)$$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}(\boldsymbol{x}), \quad \hat{\boldsymbol{y}} : \mathbb{R}^n \to \mathbb{R}^K, \quad \boldsymbol{x} \mapsto \hat{\boldsymbol{y}}$$
$$\hat{\boldsymbol{y}} : \mathbb{R}^n \to \mathbb{R}^d \to \mathbb{R}^K, \quad d \gg n, K$$

$$h = f(\boldsymbol{W}_h \boldsymbol{x} + \boldsymbol{b}_h)$$
$$\hat{y} = g(\boldsymbol{W}_y \boldsymbol{h} + \boldsymbol{b}_y)$$

Neural network (training I)

logits: output of final layer

$$\mathrm{soft}(arg)\max(\boldsymbol{l})[c] \doteq \frac{\exp(\boldsymbol{l}[c])}{\sum_{k=1}^{K}\exp(\boldsymbol{l}[k])} \in (0, 1)$$

$$\mathcal{L}(\hat{\boldsymbol{Y}}, \boldsymbol{c}) \doteq \frac{1}{m}\sum_{i=1}^{m} \ell(\hat{\boldsymbol{y}}^{(i)}, c_i), \quad \ell(\hat{\boldsymbol{y}}, c) \doteq -\log(\hat{\boldsymbol{y}}[c])$$

cross entropy / negative log-likelihood

Why not use L2-norm as cost function?

$$\boldsymbol{x}, \quad c = 1 \quad \Rightarrow \quad \boldsymbol{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix} \quad \Rightarrow \quad \ell\left(\begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix}, 1\right) \to 0^+$$

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix} \quad \Rightarrow \quad \ell\left(\begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix}, 1\right) \to +\infty$$

$$h = f(\boldsymbol{W}_h x + \boldsymbol{b}_h)$$

$$\hat{y} = g(\boldsymbol{W}_y h + \boldsymbol{b}_y)$$

Neural network (training II)

$$\Theta \doteq \{\boldsymbol{W}_h, \boldsymbol{b}_h, \boldsymbol{W}_y, \boldsymbol{b}_y\}$$

$$J(\Theta) \doteq \mathcal{L}(\hat{Y}(\Theta), c) \in \mathbb{R}^+$$

$$\frac{\partial J(\Theta)}{\partial \boldsymbol{W}_y} = \frac{\partial J(\Theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \boldsymbol{W}_y}$$

$$\frac{\partial J(\Theta)}{\partial \boldsymbol{W}_h} = \frac{\partial J(\Theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial \boldsymbol{W}_h}$$

back-propagation